

FEATURE ARTICLE

Jeff Fisher

Use Infrared to Make Embedded Printing Easy

Just how cheap and easy can it be to add a printer to an embedded system? With some nifty innovation, Jeff modifies a HP calculator printer to communicate with his computer using an infrared signal.

"h

i. This is Mr. Bigbucks at Monolithic Megacorp. We really like your Frapdoodle 2000. Work us up a quote on a container-ship load right away. By the way, there's just this one thing...is there some way that it can print a log of fraps on a little printer? Remember, cost is our number one concern!"

"Sure, no problem. I'll have a quote for you in the morning."

Click. Groan. Data logging. Yuk. Let's see, a UART plus glue for serial data (PC board re-layout!), a DE9 connector (case modifications!), a cable to connect to the printer.... Now, where did I see those surplus cash-register printers advertised? I wonder if I can find a case for them. Oh yea, and a power supply. Groan.

"Where's the aspirin?"

Have you ever needed a little printer for your stand-alone project? We went through this recently and discovered that it is more difficult and expensive than it sounds. But instead of telling you all the reasons you can't do it, I'll tell you how some lateral thinking may achieve the desired results in an inconspicuous way.

CALCULATING AN APPROACH

You've seen these new personal information managers (PIMs) and high-powered calculators? Many have

optional printers. All these printers have a nonstandard method of connection. Many plug into the calculator or PIM with a custom connector and receive clocked-serial, TTL-level data.

But Hewlett-Packard uses a unique approach. Their HP82240 calculator printer (see Photo 1) receives infrared signals from the calculator. The printer has 24 columns, prints on a 2 $\frac{1}{4}$ × thermal-paper roll, can operate on internal batteries or a simple external power supply, is available anywhere that sells HP calculators, and is relatively inexpensive (around \$130).

What could be better than a wireless connection? All I need is an infrared LED poking out somewhere, a single bit from the micro to drive it, a little software, and behold the printed word! The interface cost is so low that I could build it into every unit, and then offer the printer as an option.

First, a little reverse engineering. Now, I happen to know as much about infrared as any other human being alive, which is almost enough.

Nobody ever knows "enough" about infrared. But, my tools served me well in deciphering the infrared codes, and I did eventually figure out what the heck those extra bits on each character were. Having done the hard work, it turns out that the codes are relatively easy to create.

ANATOMY OF A CHARACTER

Like most printers, the HP82240 receives eight-bit characters. The



Photo 1—The HP82240B printer—24 columns, graphics capable, battery or AC powered, wireless infrared operation.

lower 128 are standard ASCII characters. The upper 128 are special characters that include various symbols, foreign characters, accent marks, and so on. Escape codes offer expanded (double-wide) printing, underlining, and even dot-addressable graphics. (This is all explained in the printer's manual.) If you are unfamiliar with infrared, you should read the sidebar on infrared communications before going any further.

It takes 10.92 ms to transmit a character (see Figure 1a). Each character is followed by at least 4.7 ms of no carrier. A carriage return causes the printer to output the line and advance the paper, which takes about 1.8 s. Since there is no feedback from the printer, output must be paced so it doesn't overrun the meager printer buffer. (Note that when the printer is operated on battery power, it runs slower as the batteries discharge.)

Now look at Figure 1b. Each character consists of 13 bits. The start bit (always a one) is followed by four bits of longitudinal-redundancy check (LRC) and eight bits of character. The LRC and character bits are transmitted high bit first. The LRC is calculated using only the byte it is attached to.

There's probably some kind of giant polynomial expression that could be used to create the LRC, but I ignored this. Instead, I created a simple procedure using bit tests and exclusive ORs. (One of the advantages of reverse engineering is that you get to avoid all the theories and math that went into the original design.)

The bits are transmitted using a technique similar to modified frequency modulation (MFM)—that's right, the same scheme used in many disk drives! The advantage of MFM is that it is self-clocking, well understood, and reliable. Note that MFM modulates the frequency of pulses, not the frequency of the carrier. The rules of this peculiar style of MFM are:

• for a data pulse, if the data bit is one, turn the carrier on in the middle of the bit cell. Otherwise, leave the carrier off in the middle of the bit cell.

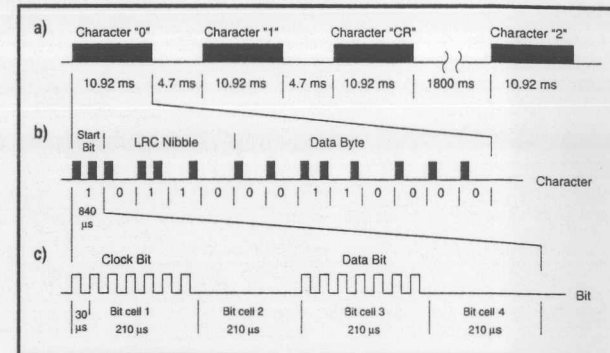


Figure 1—The transmission timing can be successively broken down into finer and finer pieces. Complete characters (a) can be divided into a series of bits (b), which are further described by bursts of IR light (c).

• for a clock pulse, if the previous bit was zero, if it is the start bit, or if the previous bit was the start bit, turn the carrier on at the start of the bit cell.

As you can see in Figure 1c, each bit cell (840 μ s) is divided into four pieces, each 210 μ s long. The second and fourth piece always have no carrier present. The third piece is the data bit and has a carrier present if the data bit is on. The first piece is where the clock pulse goes, but it is only inserted if the preceding data bit is zero.

The 33 $\frac{1}{3}$ -kHz carrier can run seven cycles in 210 μ s, which is just long enough for the detector to trigger and the main reason that the printer only works at desktop distances from the transmitter.

INFRARED LEDS

I like the SY-IR53L infrared LED available from Radio Shack (276-143). You can drive this LED satisfactorily from many TTL parts as long as you

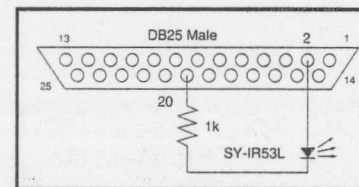


Figure 2—The only hardware required to send data to the infrared printer is a resistor and an LED connected to a bit on the PC's printer port. Software does all the work.

use an appropriate series resistor to limit the current. Alternatively, you can use some discrete parts to drive the LED closer to its 100-mA rating. Just be aware that stronger is not necessarily better in the bizarre world of infrared. I obtained successful results (over three feet) at only 2.5 mA.

So, all you need to drive a printer from your embedded system is a single TTL output bit and a processor fast enough to turn the bit on and off every 15 μ s. If you can't dedicate your processor to driving the LED during printing and 15 μ s is too fast to handle with interrupts, you can add a 33 $\frac{1}{3}$ -kHz oscillator, which you enable and disable on 210- μ s intervals. The frequency doesn't have to be precise, so a 555 or ceramic oscillator is adequate.

A SIMPLE EXAMPLE

So much for theory. Now to really do something.

For this example, we used an IBM PC-compatible computer. We connect the infrared LED and 1-k Ω series resistor to data bit zero of the parallel port (see Figure 2). Most PCs use an octal latch such as the 74LS374 for the parallel port's data bits. This part can source up to 2.6 mA, which is adequate for this example. The hardware is that simple! Software in the PC can now drive the infrared LED.

The example program, printhp (Listing 1), is written in Turbo C. It reads input and sends

\$99 4A C Compiler?

You heard right. A quality C compiler designed for the 8051 microcontroller family, just \$99, including the Intel compatible assembler and linker. And a source level simulator is also available, just \$79. A great companion to our fine Single Board Computers, like those below. **CALL NOW!**

552SBC

80C552 a '51 Compatible Micro
40 Bits of Digital I/O
8 Channels of 10 Bit A/D
3 Serial Ports (RS-232 or 422/485)
2 Pulse Width Modulation Outputs
6 Capture/Compare Inputs
1 Real Time Clock
64K bytes Static RAM
1+ UVPRAM Socket
512 bytes of Serial EEPROM
1 Watchdog
1 Power Fail Interrupt
1 On-Board Power Regulation

Priced at just **\$299** in single quantities. Call about our 552SBC C Development Kit, just \$449.

100 MHz 8051!

Our popular 8031SBC can now be shipped with Dallas Semi's hyperactive **DS80C320**, an 8051 on steroids. Averaging 3x faster than the standard 51, your project can really scream! **Call or ftp for pricing and brochures today!**

Other versions of the 8031SBC have processors with on-chip capture registers, EEPROM, IIC, A/D and more. Call or ftp for a list!

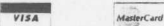
8031SBC as low as \$49

Call for your custom product needs. Quick Response.

HTE HTEch Equipment Corp.
9400 Activity Road
San Diego, CA 92126
(Fax: (619) 530-1458)

Since 1983

(619) 566-1892



Internet e-mail: info@hte.com
Internet ftp: ftp.hte.com

INFRARED COMMUNICATIONS

The most confusing part about infrared data transmission is that there are three frequencies involved.

The highest frequency, more appropriately called *wavelength*, is the infrared light emitted by the LED. For most consumer remote controls (and this calculator printer), the wavelength is around 880 nm. This wavelength is in the *near infrared* band—somewhere between visible light and actual heat radiation. You should choose an infrared LED that emits in a range of 850-900 nm.

The next highest frequency is that of the *carrier*. The carrier more reliably detects the infrared turning on and off at a known frequency than trying to detect a steady state on or off. This makes sense since *everything* emits infrared energy, so the background level is constantly going up and down. Typical carrier frequencies range from 20 kHz to 80 kHz with 40 kHz being most common. Since the receiver circuits are pretty simple, the carrier can often vary by as much as 20% and still works.

The next lowest frequency is used for the pulses that carry the actual information. Most infrared data is transmitted by turning the carrier on and off at times determined by the data. This method results in pulses of the carrier, followed by no carrier, and is called *Pulse Code Modulation* (PCM).

Both the on and off times can vary to carry data, so the frequency of the pulses is not necessarily constant. Usually, only the on or off time is varied to keep things simple. Since it may take many cycles for the receiver to react to the carrier, the on times are usually at least ten times longer than the carrier wavelength. Thus, the pulses are usually measured in hundreds of microseconds of on and off time.

Listing 1—This sample program demonstrates printing on the HP82240B infrared printer. It reads from standard input.

```
#include <stdio.h>
#include <dos.h>

/***** STATIC VARIABLE DECLARATIONS *****/
#define LPT1 0x378
#define LPT2 0x278
#define LPT LPT1 /* which printer port to use */
#define THIRTY 6 /* adjust to get 33.3-kHz carrier */
#define TWOTEN 55 /* adjust to get 210 µs delay */

static unsigned short counter;

/***** STATIC FUNCTION DECLARATIONS *****/
static void printhp(char);
static void wiggle(void);
static void mydelay(void);

/***** Main function *****/
int main(int argc, char *argv[])
{
    char c;
    delay(0); /* initialize the delay system */
    outp(LPT + 2, 0); /* enable output on parallel port */

    while ((c=getchar()) != EOF) {
        printf("%c", c);
    }
}
```

(continued)

Listing 1—continued

```
printhp(c);
delay(5); /* intercharacter gap */
if (c == '\n')
    delay(1800); /* add about 1.8 seconds delay after CR */
return 0;

/***** Print a single character on the HP infrared printer *****/
void printhp(char c)
{
    unsigned int i, j;
    static int lastbit;

    i = c; /* put in the character */
    if (i & 0x01) i ^= 0x300; /* put in the check nybble */
    if (i & 0x02) i ^= 0x500;
    if (i & 0x04) i ^= 0x600;
    if (i & 0x08) i ^= 0x900;
    if (i & 0x10) i ^= 0xa00;
    if (i & 0x20) i ^= 0xc00;
    if (i & 0x40) i ^= 0xe00;
    if (i & 0x80) i ^= 0x700;
    i |= 0x1000; /* put in the start bit */

    /* rotate it all out: start-bit, check-nybble, char */
    for (j = 0, lastbit = 0; j < 13; j++, i <<= 1) {
        if (lastbit)
            mydelay(); /* no clock pulse needed */
        else
            wiggle(); /* clock pulse */
        mydelay(); /* wait until middle of bit cell */
        if (i & 0x1000)
            wiggle(); /* on bit */
        else
            mydelay(); /* off bit */
        mydelay(); /* fourth part of bit cell */
        if (j)
            lastbit = i & 0x1000; /* save bit for clocking next pass */
    }

    /***** Delay for 210 µs *****/
    static void mydelay(void)
    {
        counter = 0;
        while (counter++ < TWOTEN);
    }

    /***** Wiggle the output bit 7 times to create a carrier *****/
    static void wiggle(void)
    {
        int i;
        for (i = 0; i < 7; i++) {
            counter = 0;
            outp(LPT, 0xff); /* turn the bit on */
            while (counter++ < THIRTY/2); /* wait 15 µs */
            outp(LPT, 0x00); /* turn the bit off */
            while (counter++ < THIRTY); /* wait 15 µs */
        }
    }
}
```

each character to then disconnect is an meant to be used as filter. To make the mand-line pipe (e.g., type `a-v, re-printhe`). The main routine handles the reading of characters and gross timing issues. The routine `printhe` encodes and outputs the characters. The `mydelay` routine pauses for 210 µs, and `wiggle` creates a brief 33.333-kHz carrier pulse.

If you want to run `printhe`, you need an oscilloscope or logic analyzer to calibrate the two routines. First, get the carrier cycle length as close to 30 µs as possible by changing the `THIRTY` definition. Then, adjust `TWOTEN` so the gap between the first two carrier pulses is as close as possible to 210 µs. This should get your printer working.

THE NEXT DAY

"Mr. Bigbucks again. Thanks for the quote and the demonstration of your new FrapLog option last week. But, uh, about the order...Monolithic Megacorp was just bought out by Polythitic Gigacorp. It seems that Polythitic just don't give a frap..."

Click. Oh, well. At least some of our real customers will appreciate it. ☺

Jeff Fisher is president of HomeTech Solutions, a home automation manufacturer and retailer in San Jose, California. He may be reached at (408) 257-4406 or 71431.3343@compuserve.com.

SOURCES

The HP82240B printer is available from:

HomeTech Solutions
10570 S. De Anza Blvd.
Cupertino, CA 95014
(408) 257-4406
Fax: (408) 257-4389

Hewlett-Packard
Portable Computer Division
1000 NE Circle Blvd.
Corvallis, OR 97330
(503) 757-2000

IRS

404 Very Useful
405 Moderately Useful
406 Not Useful